Discrete Applied Mathematics 39 (1992) 197–205
North-Holland

Openshop scheduling with machine dependent processing times

Moshe Dror

Decision Sciences, College of Business, The University of Arizona, Tucson, AZ 85721, USA

Received 6 December 1989
Revised 3 December 1990

Abstract

Dror, M., Openshop scheduling with machine dependent processing times, Discrete Applied Mathematics 39 (1992) 197–205.

This paper examines the openshop problem with machine dependent processing times. Two objectives are considered; minimizing the maximal completion (makespan), and minimizing the mean flow time. For this problem with the makespan criteria and the number of jobs greater or equal to the number of machines we present an O(mn) optimal algorithm and prove that the same problem with the number of jobs less than the number of machines but greater or equal three is NP-hard. We also present an O(n) optimal algorithm for this problem with mean flow time criteria but two machines only, and for a special case with m machines describe an optimal O(mn) algorithm. The three machine openshop problem with machine dependent processing times and mean flow time criteria remains open.

1. Introduction

An openshop machine scheduling problem is defined as follows: Given a finite set $J=\{J_1, J_2, \ldots, J_n\}$ of n jobs to be processed on the set $M=\{M_1, M_2, \ldots, M_m\}$ of m machines. Job J_i, $i=1,\ldots,n$ consists of m operations $(O_{i1}, O_{i2}, \ldots, O_{im})$. Operation O_{ij} has a processing requirement of $p_{ij}\geq 0$ implying that the job J_i has to be processed (uninterrupted if preemption is not allowed) for p_{ij} time units on machine M_j. In an openshop, the order in which the operations of a job are processed is immaterial. For the sake of completeness we define a flowshop and a permutation flowshop. In a flowshop the machines are ordered and processing of the operation O_{ij+1} may start only after the operation O_{ij} has been completed. A permutation flowshop is a flowshop where all the machines process the jobs in the same order.

Correspondence to: Professor M. Dror, Decision Sciences Group, MIS Department, University of Arizona, Tucson, AZ 85721, USA.

Common constraints on most machine scheduling problems are that two operations of the same job cannot be processed at the same time and that a machine cannot process more than one job at a time.

A machine schedule specifies the order and time each job J_i (each operation O_{ij}) $i=1,\ldots,n$ is processed on each machine M_j, $j=1,\ldots,m$ while satisfying the scheduling constraints. A completion time C_i for job J_i in a given job schedule is the sum of its processing times ($\sum_{j=1}^{m} p_{ij}$) plus all the idle time before the start of its processing on the last machine. (We assume that all the jobs are available at time 0.) A makespan problem is that of determining a job schedule which minimizes the maximal completion time for the set J. A mean flow time problem is the problem of determining a machine schedule which minimizes the sum of completion times. Following the three field notation $\alpha|\beta|\gamma$ from Graham et al. [9] we denote by $O\|C_{\max}$ an openshop problem with the objective of minimizing the maximal completion time. The problem $O|p_{ij}=1|C_{\max}$ denotes the openshop makespan problem with equal processing times for all jobs on all machines. In terms of complexity classification (see Garey and Johnson [7] for definition of terms and a detailed discussion), for many variants of the openshop machine scheduling problem, the issue as to whether a given problem is NP-hard or "belongs" to P, has been settled. We review this classification for a number of openshop problems, in order to provide an appropriate perspective for the problems examined in this paper. For a more recent survey of scheduling theory see [6]. Most of the complexity results which we cite are taken from [9, 1, 3, 2, 10]. We start with the simplest of the openshop problems $O2\|C_{\max}$ (two machine makespan). This problem can be solved in linear time, i.e., O(n) complexity. This result, however, cannot be extended, and already $O3\|C_{\max}$ is an NP-hard problem. The problems $O2|r_j|C_{\max}$, $O2|\text{tree}|C_{\max}$, and $O\|C_{\max}$ are all NP-hard in the strong sense (r_j denotes release time for J_j, tree denotes precedence relation in set J). When the objective function is mean flow time, even the two machine case is NP-hard in the strong sense ($O2\|\sum C_i$, and also $F2\|\sum C_i$). In the case of the unit time processing requirements for each job on each machine, Adiri and Amit [3] describe efficient O(mn) algorithms for the $O|p_{ij}=1|C_{\max}$ and $O|p_{ij}=1|\sum C_i$ problems. In an article by Adiri and Aizikowitz [2] the analysis of the openshop is extended to the case of dominated machines, and a number of new complexity results are presented, including an O(n) algorithm for the $O3|\text{dominated machine}|C_{\max}$ problem.

2. Machine dependent openshop scheduling: makespan

In this paper we first examine the makespan openshop problem with machine dependent processing times, i.e., $p_{ij}=p_j$ for $i=1,\ldots,n$ and $j=1,\ldots,m$. In this case we assume that $p_1\geq p_2\geq\cdots\geq p_m$. For $n\geq m$ we present an O(mn) optimal algorithm similar but not the same as in [3]. For $n\geq 3$, $n<m$, we prove that this problem is NP-hard and for $O|p_{ij}=p_j,\ n=2,\ m>2|C_{\max}$ we present an optimal O(m) algo-

rithm. Following Adiri and Hefetz [4] these results further delineate the borderline complexity for the openshop and can be used in conjunction with group technology concepts to construct heuristic schedules [5].

Optimal "rotation schedule" for $O|p_{ij}=p_j,\ n\geq m|C_{\max}$.

Algorithm 2.1. Schedule all the jobs on M_1 in order. On machine M_j $(2\leq j\leq m)$ start with job j in order till job n, then jobs 1 till $j-1$ in order.

Optimality of Algorithm 2.1 is obvious, since the lower bound on $C_{\max}$ is obtained on M_1. The time complexity of this algorithm is O(mn).

Next, we prove a counterintuitive result, that if the number of machines m is greater than the number of jobs in the system, but the number of jobs is at least three, the above makespan problem is NP-hard.

Theorem 2.2. *The problem* $O|p_{ij}=p_j,\ n\geq 3,\ n<m|C_{\max}$ *is* NP-*hard.*

Proof. Examine the special case $n=3$ and $n<m$. We first state that this problem is equivalent to $O3|p_{ij}=p_i|C_{\max}$ ($p_{ij}=p_i$ implies that a job requires an identical processing time on each machine) where the number of jobs is greater than the number of machines.

We view each job in the $O|p_{ij}=p_j|C_{\max}$ problem as if it were a "machine" and each machine as if it were a "job" which requires processing on each of the corresponding "machines". Thus, we obtain an equivalent problem of $O|p_{ij}=p_i|C_{\max}$ and in particular $O|p_{ij}=p_j,\ n=3,\ m>3|C_{\max}$ is equivalent to $O3|p_{ij}=p_i,\ n>m|C_{\max}$.

In proving that $O3|p_{ij}=p_i,\ n>m|C_{\max}$ is NP-hard we make use of the following NP-complete problem PARTITION. A multiset $S=\{a_1,\ldots,a_n\}$ is said to have a partition if there exists a subset $U\subset\{1,2,\ldots,n\}$ such that

$$\sum_{i\in U} a_i = B \quad \text{where} \quad \sum_{a_i\in S} a_i = 2B.$$

The PARTITION problem is that of determining for an arbitrary multiset S whether it has a partition. The a_i may be assumed integer.

We prove that if our $O3|p_{ij}=p_i|C_{\max}$ is polynomially solvable, then so is PARTITION.

From the PARTITION problem for $S=\{a_1,a_2,\ldots,a_n\}$ construct the following instance of the openshop problem with $n+1$ jobs, $m=3$ machines and $p_{ij}=p_i$ for $1\leq i\leq n+1$, $1\leq j\leq 3$. The processing times p_i are defined as follows: $p_i=a_i$, $1\leq i\leq n$ and $p_{n+1}=B$. Now we show that the above openshop problem has a schedule with $C_{\max}=3B$ iff S has a partition. (This part is very similar to that in [8, Lemma 4.1].)

(a) If S has a partition U, then there is a schedule with $C_{\max}=3B$. Such a schedule is shown in Fig. 1.

(b) If S has no partition, then all schedules for our openshop problem must have

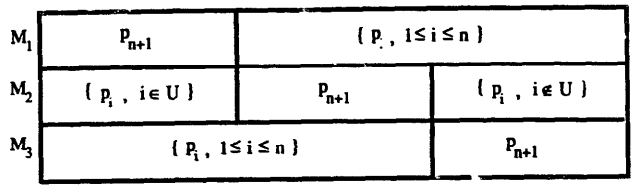

Fig. 1. Optimal schedule when S has a partition.

a completion time greater than $3B$. This is shown by contradiction. Assume that there is a schedule for the openshop problem with $C_{\max}=3B$. Since no job can be processed simultaneously on two machines, the job J_{n+1} has to be processed at all times. Given that the schedule is nonpreemptive, there must be a machine M_j on which the job J_{n+1} starts processing at time B and is completed at time $2B$. Since we assume that $C_{\max}=3B$, on machine M_j a subset of jobs has to be processed between 0 and time B and the rest of the jobs are processed from $2B$ till $3B$ on M_j. But this would constitute a partition for S.

Clearly, $n\geq 3$, thus the number of jobs in the $O3|p_{ij}=p_i|C_{\max}$ problem is ≥ 4. □

Theorem 2.2 was stated in terms of machine dependent processing times with number of jobs less than the number of machines but greater than two. In the process of proving that the problem $O|p_{ij}=p_j$, $n\geq 3$, $n<m|C_{\max}$ is NP-hard we also proved that $O3|p_{ij}=p_i$, $n>m|C_{\max}$ is NP-hard.

To complete this complexity analysis we note that $O|p_{ij}=p_j$, $n=2$, $n<m|C_{\max}$ can be solved in O(m) time, since the equivalent problem $O2|p_{ij}=p_i|C_{\max}$ is a special case of $O2\|C_{\max}$ which is solvable in O(n) time [8].

3. Machine dependent openshop scheduling: mean flow time

The next problem we examine is that of minimizing the sum of completion times in a two machine openshop with machine dependent processing times $O2|p_{ij}=p_j|\sum C_i$. We note that the general two machine case ($O2\|\sum C_i$) was proven to be NP-hard by Achugbue and Chin [1], and for the $O|p_{ij}=1|\sum C_i$ problem, Adiri and Amit [3] presented an optimal O(mn) algorithm.

We first observe that in case $p_1\geq 2p_2$ we can construct an optimal schedule (see

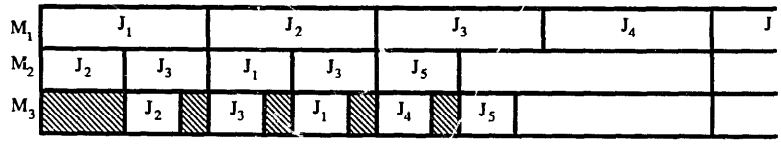

Fig. 2. Schedule according to Algorithm 3.1 with $m=3$.

Algorithm 3.1 below) in $O(n)$ time and extend it to a special case with $m \geq 3$ in $O(mn)$ time where

$$p_1 \geq 2p_2 \geq 2p_3 \geq \cdots \geq 2p_m \text{ and } p_1 \geq \sum_{i=2}^{m} p_i \text{ (equivalently } p_2 \geq \sum_{i=3}^{m} p_i).$$

Algorithm 3.1. Schedule continuously job J_1 on $M_1, M_2, \ldots, M_m$. Schedule continuously J_i, $2 \leq i \leq n$ on $M_2, \ldots, M_m$ and then on M_1. All schedules are left adjusted on M_i, $i = 1, \ldots, m$ (i.e., the jobs start as soon as possible).

Claim 3.2. *For the problem* $O \mid p_{ij} = p_j$, $p_1 \geq 2p_2 \geq 2p_3 \geq \cdots \geq 2p_m$, $\sum_{i=2}^{m} p_i \leq p_1$, $\mid \sum C_i$, *Algorithm* 3.1 *constructs an optimal schedule.*

Proof. Job J_1 is completed in $C_1 = \sum_{i=1}^{m} p_i$. Job J_i is completed in $C_i = ip_1$, $2 \leq i \leq n$. Clearly no better schedule is possible (see Fig. 2). □

We extend the result obtained in Claim 3.2 to the two machine case where the processing time on machine M_1 is greater than that on M_2 but less than two times the processing time on M_2 (i.e., $p_2 < p_1 < 2p_2$). We present an optimal $O(n)$ algorithm for this case, thus, completing the proof that the decision version of the mean flow time two machine openshop problem with machine dependent processing times belongs to P.

Following a comment by Kubiak we assume that the input for

$$O2 \mid p_{ij} = p_j,\ p_2 < p_1 < 2p_2 \mid \sum_{i=1}^{n} C_i$$

is represented as a list of n pairs (p_1, p_2), one for each job which takes $O(n(\log p_1 + \log p_2))$ bits.

Theorem 3.3. *The problem* $O2 \mid p_{ij} = p_j, p_2 < p_1 < 2p_2 \mid \sum C_i$ *can be solved optimally in* $O(n)$ *time.*

Proof. Initially assume a large number of jobs.

Algorithm 3.4. Machines M_1 and M_2 are operating continuously with no idle time for periods np_1 and np_2 respectively.

Start with machine M_1 and job J_1. Assign an uncompleted job with the lowest

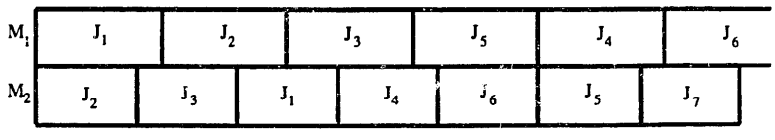

Fig. 3. Schedule according to Algorithm 3.4 ($p_2 = (4/5)p_1$).

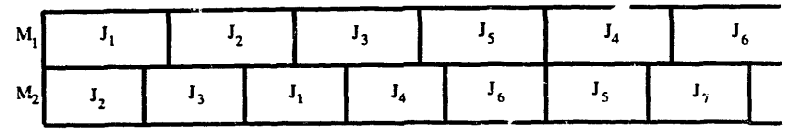

Fig. 4. Schedule produced by Algorithm 3.4 for $p_2=(4/5)p_1$.

index on the first free machine, given that it requires processing on that machine. Continue until all jobs are completed (see Fig. 3).

In order to simplify the proof for optimality of Algorithm 3.4 we partition the problem into two cases.

Case 1: $p_2/p_1 \leq 3/4$.

Algorithm 3.4 produces the following completion times: $C_1=3p_2$, $C_i=ip_1$, $i=1,\ldots,n$. The lower bound on total completion times is given by the following completion times: $\underline{C}_1=p_1+p_2$, $\underline{C}_i=ip_1$, $i=2,\ldots,n$. Subsequently, we need to examine only the completion time for the job J_1 which is greater at most by $p_1/2$ than its lower bound.

Reducing the completion time of job J_1 to its lower bound would introduce an idle time on machine M_2 of $p_2/3$, but more important, it would delay the completion time of job J_3 from $3p_1$ to $4p_1$, thus adding $p_2/2$ to the total completion time obtained by Algorithm 3.4.

Case 2: $p_2>(3/4)p_1$ (note that since p_1, p_2 are assumed integers, p_1-p_2 is an integer ≥ 1).

First we note (without a proof) that for any large but finite p_1 and p_2, such that $p_1=p_2+1$, the solution obtained by Algorithm 3.4 is superior, for sufficiently large n, to the schedule (Adiri and Amit [3]) which assumes equal p_1 and p_2. However, for a small n, the schedule produced by the algorithm of Adiri and Amit pretending $p_1=p_2$ might have a total completion time smaller than that produced by Algorithm 3.4. Let's first examine (for large n) the schedule produced by Algorithm 3.4 for $p_2=(3/4)p_1+\varepsilon$, given $\varepsilon>0$ and such that $p_1-p_2\geq 1$ and integer. This integrality of p_1, p_2 imposes a lower bound on the value of ε in terms of p_2 (or p_1). Using simple arithmetic we get $\varepsilon\geq(p_2-3)/4$ and for any k such that $k>4p_2/(p_2-3)$ we get the completion times $C_k=kp_1$, $\lceil 4p_2/(p_2-3)\rceil \leq k\leq n$ (it is the lower bound on completion time). For $j<\lceil 4p_2/(p_2-3)\rceil$ the completion times for jobs $J_1, J_2, \ldots, J_j$, respectively, are: $3p_2$, $2p_1$, $3p_1$, $5p_1$, $6p_2$, $6p_1$, $8p_1$, $9p_2$, $9p_1$, $11p_1$, $12p_2$, $12p_1$, $14p_1$, $15p_2$, etc. with this repeating pattern until C_j. Let us assume that $p_2=(4/5)p_1$ and prove the optimality of Algorithm 3.4 for that case. A similar proof can be constructed for any other specific ratio $1>p_2/p_1>3/4$.

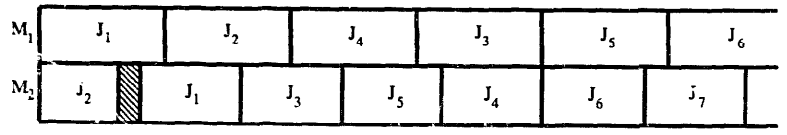

Fig. 5. Schedule produced by completing job J_1 as soon as possible.

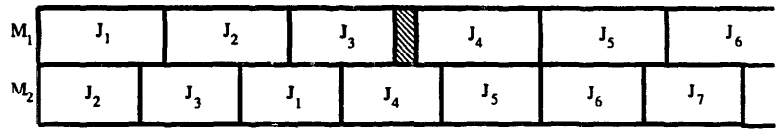

Fig. 6. Schedule produced by completing job J_4 as soon as possible.

Given $p_2=(4/5)p_1$, Algorithm 3.4 gives the following completion times for jobs $J_1, J_2, \ldots, J_n$, respectively: $3p_2$, $2p_1$, $3p_1$, $5p_1$, $6p_2$ and ip_1 for $i=6,7,\ldots,n$ (see Fig. 4).

When comparing the completion times in the above schedule with the lower bounds on the completion time for each job, we find two jobs J_1, J_4, completed late and job J_5 completed earlier. We examine in turn three schedules which manipulate the completion times of the "late" jobs. Those three candidate schedules dominate any other potential schedules.

In the first schedule, we move the completion time of job J_1 to its lower bound but then retain the schedule obtained by Algorithm 3.4 (see Fig. 5).

The difference in total completion time between the above schedule and that of Algorithm 3.4 is $(6/5)p_1$.

The second schedule attempts to complete job J_4 as soon as possible in the original schedule (see Fig. 6).

It is clear from the above schedule that the difference in total completion time between the above schedule and the original schedule increases with n.

The third schedule examined attempts to complete all the jobs at their respective lower bound on completion time except job J_1 which is completed on machine M_2 at kp_1 such that $k(p_1-p_2)=p_2$ which in case of $p_2=(4/5)p_1$ gives $k=4$ (see Fig. 7).

The difference in the total completion time between the above schedule and that of Algorithm 3.4 is $(4/5)p_1$.

As we noted before, for small n, Algorithm 3.4 does not necessarily produce the optimal schedule, but for n such that $(n-1)/n \geq p_2/p_1$ we proved that it does. In the case of a small number of jobs (which in the extreme case implies $n<p_1$) one can simply test a number of candidate schedules. □

Open problem. $O3 \mid p_{ij}=p_j \mid \sum C_j$.

4. Conclusions

In this paper we examined the openshop scheduling problem with machine depen-

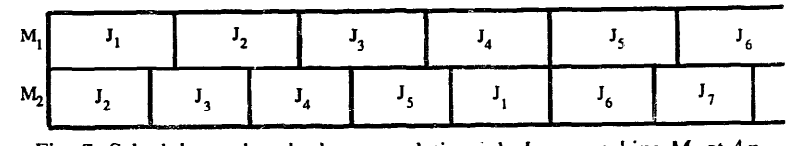

Fig. 7. Schedule produced when completing job J_1 on machine M_2 at $4p_1$.

dent processing times focussing on two criterias: the makespan and the mean flow time. For the makespan criterion with number of jobs equal or greater than the number of machines, we provided two optimal O(mn) algorithms very similar to the algorithms by Adiri and Amit [3] for the unit processing time case. In addition, we have proven that the problem $O|p_{ij}=p_j,\ n\geq 3,\ n<m|C_{\max}$ is NP-hard by reduction from PARTITION. This result is counterintuitive and further delineates the border between P- and NP-complete problems. As a by-product we have also proven that the $O3|p_{ij}=p_i,\ n>m|C_{\max}$ problem is NP-hard since the problem $O2|p_{ij}=p_i,\ n>m|C_{\max}$ is solvable in O(n) time (as a special case of $O2\|C_{\max}$) we have proven that its equivalent problem of $O|p_{ij}=p_j,\ n=2,\ n<m|C_{\max}$ can be solved in O(m) time. For the mean flow time criterion we have extended the present complexity classification for these problems by proving that:

(a) the problem $O|p_{ij}=p_j,\ p_1\geq 2p_2\geq 2p_3\geq\cdots\geq 2p_m,\ \sum_{i=2}^{m} p_i\leq p_1\,|\,\sum_{i=1}^{n} C_i$ can be solved optimally in O(mn) time,

(b) the two machine problem $O2|p_{ij}=p_j|\sum_{i=1}^{n} C_i$ can be solved optimally in O(n) time (Claim 3.2 and Theorem 3.3). For both of these cases the appropriate optimal algorithms are described.

We conclude the paper by stating an open problem which requires further research.

Acknowledgement

I would like to thank Professor I. Adiri whose research and comments inspired this work, and Dr. C. Sriskandarajah for a fruitful discussion in an early stage of this research. I am especially grateful to Dr. W. Kubiak for many critical remarks which were incorporated into the paper.

References

[1] J.O. Achugbue and F.Y. Chin, Scheduling the openshop to minimize mean flow time, SIAM J. Comput. 11 (1982) 709-720.

[2] I. Adiri and N. Aizikowitz (Hefetz), Openshop scheduling problems with dominated machines, Naval Res. Logist. 36 (1989) 273-281.

[3] I. Adiri and N. Amit, Openshop and flowshop scheduling to minimize sum of completion times, Comput. Oper. Res. 11 (1984) 275-284.

[4] I. Adiri and N. Hefetz, Subproblems of openshop—more than two machines—schedule length problems, Mimeograph Series No. 260, Faculty of Industrial Engineering and Management, Technion (1980).

[5] D. Ben-Arieh and M. Dror, Group technology for difficult groups by scheduling measures, Internat. J. Comput. Integrated Manufact. 2 (1989) 186-193.

[6] J. Blazewicz, Selected topics in scheduling theory, in: Annals of Discrete Mathematics 31 (North-Holland, Amsterdam, 1987) 1-60.

[7] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA, 1979).

[8] T. Gonzalez and S. Sahni, Openshop scheduling to minimize finish time, J. ACM 23 (1976) 665-679.

[9] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of Discrete Mathematics 5 (North-Holland, Amsterdam, 1979) 287-326.

[10] W. Kubiak, C. Sriskandarajah and K. Zaras, Openshop scheduling: A survey, GERAD-G-89-19, Montreal, Que. (1989).